한국자원공학회지 J. Korean Soc. Miner. Energy Resour. Eng.
Vol. 52, No. 1 (2015) pp. 81-90, http://dx.doi.org/10.12972/ksmer.2015.52.1.081

ISSN 2288-0291(print)
ISSN 2288-2790(online)

기술보고

SU 파일 입출력을 위한 포트란 라이브러리 개발

하완수[1)]*

Development of a Fortran Library for Input/Output of SU Files

Wansoo Ha*

(*Received 28 January 2015; Final version Received 25 February 2015; Accepted 26 February 2015*)

Abstract : This paper suggests a Fortran library for reading, editing, and writing SU files of the Seismic Un*x data processing package. The basic input/output routines are developed in C to support SU binary formats and the Fortran module calls those routines using ISO_C_BINDING module of Fortran 2003. The Fortran interface is designed in the object-oriented way so that users can access SU files and traces using corresponding class instances. Code examples demonstrate that researchers and students studying seismic data processing using Fortran can easily read and write SU files.

Key words : Seismic Un*x, SU files, Fortran, Input/Output, Seismic data processing

요 약 : 본 연구에서는 포트란 언어 사용자들이 Seismic Un*x 자료 처리 패키지에서 사용하는 SU 형식의 파일들을 자유롭게 읽고, 수정하고, 쓸 수 있도록 입출력 라이브러리를 개발하였다. SU 이진 형식 지원을 위해 기본 입출력 함수들은 C로 구현한 후 포트란 2003의 ISO_C_BINDING 모듈을 이용하여 포트란에서 해당 함수들을 사용할 수 있도록 하였다. 포트란 라이브러리는 객체 지향적으로 설계하여 사용자들이 SU 파일과 트레이스에 해당하는 클래스 객체들을 통해 파일과 트레이스에 접근할 수 있도록 하였다. 본 라이브러리를 이용하면 포트란을 이용해 탄성파 자료 처리를 연구하는 연구자들과 학생들이 쉽게 SU 파일을 읽고 쓸 수 있음을 예제들을 통해 보였다.

주요어 : Seismic Un*x, SU 파일, 포트란, 입출력, 탄성파 자료 처리

서 론

탄성파 탐사시 취득하여 디스크에 기록하는 자료는 기본적으로 수신기에서 기록하는 이산 시계열 자료이다. 그러나 후속 처리를 위해서는 시계열 자료 외에 다양한 정보들이 필요하다. 예를 들면, 자료의 샘플링 간격, 샘플 개수, 송신기 번호, 수신기 번호, 송신기와 수신기의 좌표 등의 정보가 있다. 일반적으로 이러한 정보들은 탄성파 자료의 헤더(Header)라 부르는 부분에 저장한다.

헤더에 들어가는 정보의 종류나 헤더 및 시계열 자료의 구조는 탐사의 목적이나 종류에 따라 달라진다. 탄성파 탐사에서 많이 사용하는 형식으로는 SEG-D(Society of Exploration Geophysicists - D format) 및 SEG-Y (Society of Exploration Geophysicists - Y format) 형식 (SEG Technical Standards Committee, 2002; SEG Technical Standards Committee, 2012)이 있다. SEG-D는 주로 다중 채널 반사법 탐사시 사용되며, SEG-Y 형식은 그 외의 탐사나 자료의 처리 및 저장시에 널리 이용된다. 3차원 육상 탐사의 위치 정보 기록을 위해서는 SPS(Shell Processing Support Format) 형식(SEG Technical Standards Committee, 2006)을 많이 사용하며, 해상 탐사의 위치 정보 기록을 위해서는 P1/90, P2/94 및 후속 형식들(UKOOA Exploration Committee, 1990; International Association of Oil & Gas Producers, 2012)을 많이 사용한다. 이러한 형식들은 기본적으로 자료 취득을 위한 형식들이고, 구조가 복잡하다. 따라서 자료의 처리를 위해서는 위의 자료 형식들을 그대로 사용하기보다는 좀 더 다루기 쉬운 형식으로 변환하여 사용하는 경우가 많다.

자료 처리를 위한 형식은 자료 처리 패키지에 따라 다

1) 부경대학교 에너지자원공학과
*Corresponding Author(하완수)
E-mail; wansooha@pknu.ac.kr
Address; Dept. of Energy Resources Engineering, Pukyong National University, Busan, Korea

양한 형식이 존재한다. 상용프로그램인 ProMax의 경우 Bricked Seismic 형식, Omega에서는 WesternGeco DIO 형식을 사용한다. 공개 프로그램인 Seismic Un*x에서는 SEG-Y 형식을 수정한 SU 형식(Stockwell, 1999; Stockwell and Cohen, 2008), SEP(Stanford Exploration Project) 라이브러리에서는 SEP 형식(Clapp *et al*., 2004), Madagascar에서는 SEP 형식과 유사한 RSF 형식(Madagascar, 2015)을 사용한다. 이러한 자료 처리 형식들 중 관련 연구자들 사이에서 최근 가장 널리 사용되는 형식으로는 SU 형식과 RSF 형식을 들 수 있다(Sheen *et al*., 2003; Xiang *et al*., 2011).

자료 처리 알고리즘을 연구하는 연구자 또는 학생의 경우 직접 개발한 프로그램에서 위에 제시된 자료 형식을 사용하면 몇 가지 장점을 얻을 수 있다. 우선, 다른 연구자와 파일을 주고받을 때 표준화된 자료 형식을 사용하기 때문에 시스템이 달라도 문제가 없다. 또한 해당 자료 형식을 수정, 처리하거나 영상화하는 프로그램들이 패키지에 존재하기 때문에 이미 존재하는 기능을 새로 개발할 필요가 없다. 그러나 이 경우 패키지에서 자신이 사용하는 프로그래밍 언어를 지원하는가 하는 문제가 발생하게 된다.

Seismic Un*x와 SEP 라이브러리 및 Madagascar는 모두 기본적으로 C언어로 작성되어 있다. C언어는 유닉스 또는 리눅스 운영체제와 밀접하게 연관되어 있어 복잡한 자료 구조, 자료형 변환과 시스템 입출력 등을 다루기에 용이하다. 따라서 C언어를 사용하면 SU 형식이나 SEP/RSF 형식을 쉽게 사용할 수 있다. 그러나 연구자 중 포트란 언어 사용자가 많음을 고려하여 SEP 라이브러리와 Madagascar에서는 포트란 입출력을 위한 응용 프로그램 인터페이스(Application Programming Interface, API)를 제공한다(Madagascar, 2015). 반면에 Seismic Un*x에서는 C언어 외에 별도의 언어를 위한 인터페이스를 제공하지 않는다(Stockwell and Cohen, 2008). 따라서 본 연구에서 포트란 언어 사용자들이 SU 파일을 읽고 쓸 수 있도록 포트란 입출력 라이브러리를 개발하였다.

본 기술보고에서는 우선 SEG-Y 및 SU 파일의 구조와 SU 파일의 이진 입출력을 위해 고려해야 할 사항들을 살펴본다. 이후 새로 개발한 객체지향 포트란 SU 입출력 라이브러리의 계층과 클래스 구조를 설명하고 예제 프로그램들을 제시한다.

SU 파일

SU 파일의 구조는 널리 사용되는 SEG-Y 파일의 구조와 비교하여 이해하는 것이 좋다. 따라서 우선 SEG-Y 파일과 SU 파일의 구조를 살펴보고 파일 입출력을 위한 이진 형식에 대해 살펴본다.

SEG-Y 파일 구조

SEG-Y 파일은 Fig. 1과 같이 라벨, 파일 헤더들과 트레이스들로 이루어져 있다(SEG Technical Standards Committee, 2002). SEG-Y 테잎 라벨은 128 바이트의 아스키(ASCII, American Standard Code for Information Interchange) 형식으로, 선택적으로 파일 앞부분에 추가할 수 있다. 텍스트 파일 헤더는 3200 바이트의 EBCDIC (Extended Binary Coded Decimal Interchange Code) 또는 아스키 형식이며, 총 400 바이트의 이진(Binary) 파일 헤더는 2바이트와 4바이트 크기의 정수들로 이루어져 있다. 확장 텍스트 파일 헤더는 텍스트 파일 헤더와 같은 크기로, 여러 개가 존재할 수 있다. 확장 텍스트 파일 헤더의 개수는 이진 파일 헤더 내에 저장하여 SEG-Y 파일을 읽을 때 어디까지가 확장 텍스트 파일 헤더인지 판단할 수 있다.

파일 헤더들 이후에는 트레이스들이 이어진다. 각각의 트레이스들은 트레이스 헤더와 트레이스 자료로 구성된

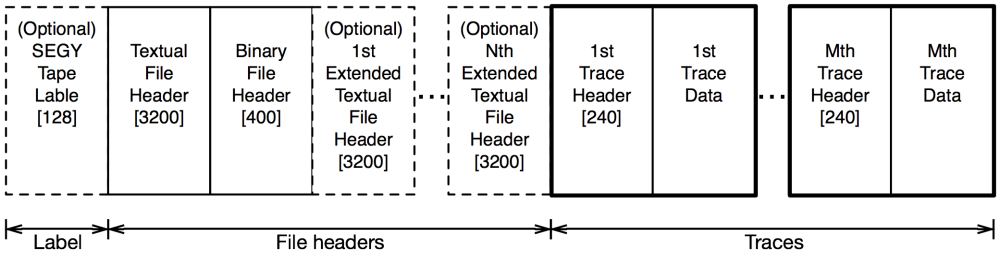

Fig. 1. The structure of a SEG-Y file format (SEG Technical Standards Committee, 2002). Numbers in brackets show the size of the structure in bytes. The size of a SEG-Y trace data can vary depending on the binary format and the number of samples.

Table 1. Available binary formats of SEG-Y trace data

Format	Element size (bytes)	Format code in the binary file header
IBM floating-point	4	1
Two's complement integer	4	2
Two's complement integer	2	3
IEEE floating-point	4	5
Two's complement integer	1	8

다. 트레이스 헤더 하나의 크기는 총 240 바이트로, 기본적으로 2바이트 및 4바이트 정수들로 이루어져 있다. 트레이스 자료는 탄성파 탐사 시계열 자료를 기록한 것으로, 샘플 개수와 이진 형식에 따라 길이가 달라진다. 샘플 개수는 트레이스 헤더로부터 알 수 있고, 트레이스 자료의 이진 형식은 이진 파일 헤더로부터 알 수 있다. 사용 가능한 트레이스 자료의 이진 형식들은 Table 1에 표시하였다. 흔치 않은 경우지만, SEG-Y 표준에 의하면 샘플 개수는 트레이스별로 달라질 수 있다(SEG Technical Standards Committee, 2002). 따라서 트레이스의 길이 또한 트레이스별로 달라질 수 있다.

SU 파일 구조

SU 파일은 라벨과 파일 헤더들을 제거한, SEG-Y 파일의 트레이스와 유사한 구조로 이루어져 있다(Fig. 2). SEG-Y 파일은 Seismic Un*x 패키지의 segyread 명령

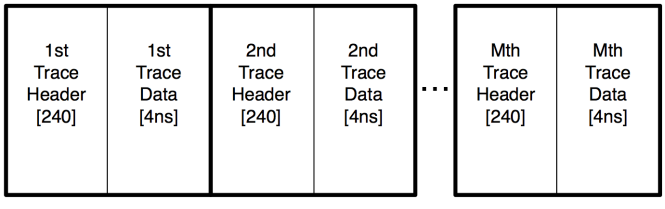

Fig. 2. The structure of an SU file. Numbers in brackets show the size of the structure in bytes, where "ns" is the number of samples in a trace.

Table 2. Keywords in the SU trace header

Location in the header (bytes)	C Type	Element size (bytes)	Keywords
1-28	int	4	tracl, tracr, fldr, tracf, ep, cdp, cdpt
29-36	short	2	trid, nvs, nhs, duse
37-68	int	4	offset, gelev, selev, sdepth, gdel, sdel, swdep, gwdep
69-72	short	2	scalel, scalco
73-88	int	4	sx,sy,gx,gy
89-114	short	2	counit, wevel, swevel, sut, gut, sstat, gstat, tstat, laga, lagb, delrt, muts, mute
115-118	unsigned short	2	ns, dt
119-180	short	2	gain, igc, igi, corr, sfs, sfe, slen, styp, stas, stae, tatyp, afilf, afils, nofilf, nofils, lcf, hcf, lcs, hcs, year, day, hour, minute, sec, timbas, trwf, grnors, grnofr, grnlof, gaps, otrav
181-204	float	4	d1, f1, d2, f2, ungpow, unscale
205-208	int	4	ntr
209-212	short	2	mark, shortpad
213-240	short	2	unass(14)

어를 이용하여 SU 파일로 변환할 수 있다. 하나의 SU 트레이스는 240바이트의 트레이스 헤더와 트레이스 자료로 이루어진다(Stockwell and Cohen, 2008). SEG-Y 파일의 트레이스와 다른 점은 트레이스 헤더의 180바이트 이후의 값들과 자료를 저장하기 위한 이진 형식이다.

SU 트레이스 헤더의 경우 총 240바이트 중 앞부분 180바이트의 구조는 SEG-Y 파일의 트레이스 헤더와 같다(Table 2). 처음 SEG-Y 파일이 발표되었을 때 트레이스 헤더의 181바이트부터 240바이트까지는 추후의 확장을 위해 비워두었는데(Barry *et al*., 1975), 이 부분에 SU 파일 자체적으로 필요한 정보를 추가하여 사용해 왔다(Stockwell and Cohen, 2008). 이후 SEG-Y 수정 형식이 발표되면서 180바이트 이후를 사용하여 SU 파일의 트레이스 헤더와 SEG-Y 파일의 트레이스 헤더에 차이가 생기게 되었다(SEG Technical Standards Committee, 2002). 따라서 헤더의 180바이트 이후에 정보를 저장한 SEG-Y 파일을 SU 파일로 변환할 경우 SU 파일의 해당 위치에 잘못된 정보가 들어가게 되고, 이 부분의 정보를 제거해주는 프로그램으로 segyclean이 있다.

SU 트레이스 자료의 이진 형식으로는 SEG-Y의 경우와 달리 4바이트 실수만 사용할 수 있다. 또한 하나의 SU 파일 내에서 트레이스별로 샘플 개수가 달라지는 것을 허용하지 않는다. 하나의 트레이스에 포함될 수 있는 최대 샘플 개수는 32767개로 지정되어 있으며, 이는 부호 있는 2바이트 정수로 표현할 수 있는 최대값과 같다(Stockwell and Cohen, 2008).

이진 형식

SU 파일의 이진 형식은 SU 파일의 구조와는 별도로, 파일의 입출력과 관련되어 있다. 트레이스 헤더와 트레이스 자료 모두 시스템 고유의 이진 형식(Native Binary) 또는 XDR(eXternal Data Representation) 형식을 지원한다. 시스템 고유의 이진 형식은 컴퓨터 프로세서나 운영체제에 따라 달라지지만, XDR 형식은 시스템 간 자료 전송을 위한 표준 형식이다(Sun Microsystems, Inc., 1987). 어떤 형식을 사용할 것인지는 Seismic Un*x 패키지 설치시의 설정에 따라 달라진다. 과거에는 시스템 고유의 이진 형식이 기본 값이었으나, 최근 배포되는 패키지에서는 XDR 형식이 기본 값으로 지정되어 있다. 이진 형식과 관련하여, 부동소수점 실수의 구조와 엔디안(Endianess)을 구분하여 살펴보아야 한다.

시스템에 따라 달라지는 부동소수점 실수의 기록 방법은 IEEE 754(Institute of Electrical and Electronics Engineers Standard for Floating-Point Arithmetic), MBF (Microsoft Binary Format), IBM 및 Cray 실수 형식 등이 있다(IEEE Computer Society, 2008). 기록 방법에 따라 단정밀도(Single precision) 및 배정밀도(Double precision)에서 지수 및 가수가 차지하는 비트수, 부호 비트의 위치 등이 달라진다. 대부분의 최신 시스템에서는 IEEE 754 표준을 따른다. XDR 형식의 경우 부동소수점 실수는 IEEE 754 표준에 따라 기록한다.

자료의 메모리 또는 파일 저장시 바이트 기록 순서는 엔디안에 따라 달라진다. 엔디안은 바이트 정렬 순서와 관련된 것으로, 최상위 비트가 앞에 오면 빅엔디안(Big endian), 최하위 비트가 앞에 오면 리틀엔디안(Little endian)이라 하며, 시스템에 따라 달라진다. SU 파일에서 시스템 고유의 이진 형식을 사용하면 시스템에 따라 빅엔디안 또는 리틀엔디안 방식이 될 수 있으나, XDR 형식은 빅엔디안으로 지정되어 있다. 일반적으로 IBM RS6000, SUN, NeXT 등의 시스템은 빅엔디안을 사용하고, 인텔 CPU를 사용하는 시스템 및 DEC 컴퓨터는 리틀엔디안을 사용한다.

시스템 고유의 이진 형식을 사용할 경우 작업하는 시스템 상에서 파일과 메모리 사이의 자료 입출력 과정에서 자료형의 변환 없이 빠르게 작업할 수 있지만, 다른 시스템으로 SU 파일을 옮길 때에는 호환성에 문제가 생길 수 있다. 반면에, XDR 형식을 사용하면 파일 입출력 시 변환 과정이 필요하지만 다른 시스템에서도 동일한 SU 파일을 사용할 수 있다. 기존의 시스템 고유 이진 형식의 SU 파일을 XDR 형식의 SU 파일로 변환하는 프로그램으로 suoldtonew라는 프로그램이 존재한다. 반대로 변환하는 프로그램은 현재 패키지에서 별도로 제공하지 않는다.

포트란 라이브러리

앞에서 살펴본 SU 파일의 입출력을 위해 필요한 기능들을 살펴보고, 본 라이브러리에서 해당 기능들을 어떻게 구현하였는지 살펴본다. 이후에는 간단한 예제들을 통해 SU 파일의 입출력 방법을 살펴본다.

입출력 함수들

SU 파일의 입출력을 위해 가장 기본이 되는 함수들은 파일을 열고 닫는 함수들과 트레이스를 읽고 쓰는 함수들이다. 별도의 구분이 필요 없을 경우 함수와 서브루틴을 모두 함수라 칭하였다. 앞서 설명한 SU 파일의 구조를 살펴볼 때 이러한 필수 함수들은 시스템 고유의 이진 파일과 XDR 파일을 모두 다룰 수 있어야 한다. 또한 Seismic Un*x 패키지의 프로그램들과 같이 유닉스/리눅스의 표준 입출력을 이용하고자 할 경우에는 파일 입출

력 뿐 아니라 표준 입출력용 함수들도 필요하다. 트레이스를 메모리상에 읽어 들인 후에는 헤더의 키(keyword)와 트레이스 자료를 읽고 수정하는 기능이 필요하다.

이외에 효율성과 편의를 위한 함수들로는 헤더만 읽는 함수들, 순차 입출력이 아니라 특정 트레이스를 읽기 위해 파일 위치 지시자를 옮기는 함수, 파일 전체 트레이스 개수를 알아내기 위해 샘플 개수와 파일 크기를 알아내는 함수 등을 들 수 있다.

이러한 함수들 중 시스템 고유의 이진 파일은 포트란 언어만으로도 다룰 수 있으나, XDR 파일과 표준 입출력을 다루기 위해서는 포트란만으로는 한계가 있다. 따라서 본 연구에서는 기본 입출력 함수들을 C로 구현한 후 포트란에서 C 함수를 불러내는 방식으로 포트란 라이브러리를 작성하였다.

포트란과 C

포트란과 C로 구성된 라이브러리의 작동을 위한 계층 구조는 Fig. 3과 같다.

포트란 2003에서는 ISO_C_BINDING이라는 모듈을 제공하여 C와 포트란 사이의 호환성을 높이고 있다 (Metcalf *et al*., 2012). 본 연구에서도 포트란에서 SU 파일 구조에 해당하는 자료형을 정의하거나 C 함수를 실행할 때 ISO_C_BINDING 모듈을 사용하여 C와의 호환성을 높일 수 있도록 하였다.

SU 파일에서 사용하는 자료형으로는 2바이트 정수, 4바이트 정수, 4바이트 실수가 있고, 이는 각각 C 자료형으로 (unsigned) short, int, float에 해당한다. 각각의 C 자료형에 대응되는 포트란 자료형은 Table 3에 제시하였다. 단, 포트란에는 C의 부호가 없는 2바이트 정수형(unsigned short)에 해당하는 자료형이 없다(Metcalf *et al*., 2012). SU 파일에서 부호가 없는 2바이트 정수형으로 정의된 헤더 값은 샘플 개수(“ns”)와 샘플링 간격(“dt”)이 있는데 (Table2), 포트란에서 부호가 있는 2바이트 정수를 사용할 경우 값이 32767이 넘으면 문제가 생긴다. 그러나, SU 파일 정의에서 최대 샘플 개수는 32767로 정해져 있고, 샘플링 간격도 일반적으로 32767 μs가 넘지 않으므로 크게 문제되지 않는다. 참고로, 샘플링 간격이 이보다 클 경우 실수로 정의된 “d1” 키를 이용할 수 있다.

객체 지향 설계

본 연구의 포트란 입출력 함수들은 사용 편의를 위해 포트란 2003에 추가된 객체 지향 프로그래밍 기능들을 이용하여 작성하였다. 포트란에서 사용하는 용어와 일반 객체 지향 프로그래밍 언어에서 사용하는 용어는 일치하지 않는데, 포트란의 사용자 정의 자료형(derived data

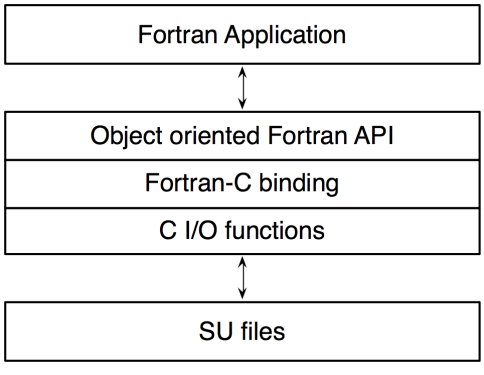

Fig. 3. The layer structure of the library.

Table 3. Data types of C and Fortran used in the SU trace

Programming Language	C Type	Fortran 90 type	Fortran 2003 type (ISO_C_BINDING module)
Data Types	int	integer (kind=4)	integer (c_int)
	short (unsigned short*)	integer (kind=2)	integer (c_short)
	float	real (kind=4)	real (c_float)

*Fortran does not have an intrinsic type corresponding to “unsigned short” type in C (Metcalf *et al*., 2012).

type)은 일반 객체 지향 언어의 클래스(class)에 해당하고, 사용자 정의 자료형 변수(variable)는 객체(object, instance), 자료형에 묶인 함수와 서브루틴(type-bound procedure)은 메소드(method), 자료형을 확장(extend)하는 것은 클래스를 상속(inherit)하는 것에 대응한다(Metcalf *et al*., 2012; Metz, 2013). 이후 설명에서는 일반적인 객체 지향 프로그래밍에서 사용하는 용어를 사용한다.

본 라이브러리의 클래스 구성은 Fig. 4와 같다. 참고로, 본 라이브러리의 단순 사용을 위해서는 내부 설계 구조를 자세히 알 필요가 없는데, 사용자를 위한 설명서는 프로그램과 함께 인터넷에 공개하였다. 클래스 구성도에서 각각의 상자는 하나의 클래스를 의미한다. 상자의 첫 번째 칸에는 클래스의 이름, 두 번째 칸에는 핵심적인 클래스 내부 변수, 세 번째 칸에는 클래스에 포함된 메소드를 표시하였다. 내부 변수와 메소드 이름 옆에는 변수의 자료형과 함수의 반환 자료형을 표시하였다. 반환 자료형이 없는 메소드는 서브루틴들이다. 각 클래스의 서브루틴들만으로 모든 기능을 실행할 수 있으나 사용 편의를 위해 반환 값을 가지는 함수들을 추가하였고, 해당 함수들은 클래스 구성도에서 회색으로 표시하였다.

SuHeader_c와 SuTrace_c는 C언어와의 통신을 위한 자료형으로, 이러한 자료형은 포트란 언어의 제한으로 인해 메소드를 가질 수 없다(Metcalf *et al*., 2012). SuTrace_c는 트레이스 헤더를 저장하기 위한 SuHeader_c와 트레이스 자료를 저장하기 위한 실수 배열을 포함한다.

SuHeader 클래스는 트레이스 헤더만을 다루기 위한 클래스로, SuTrace_c를 포함한다. SuHeader 클래스의 객체를 통해서는 헤더 정보에만 접근이 가능한데, get 메소드를 이용하여 헤더 값을 가져오고 set 메소드를 이용하여 헤더 값을 수정할 수 있다. 트레이스 자료에 접근하기 위해서는 SuHeader 클래스를 상속하는 SuTrace 클래스를 이용한다. SuTrace 클래스의 객체를 이용하면 get, set 메소드를 통해 헤더 값 뿐 아니라 트레이스 자료도 읽거나 수정할 수 있다. 메소드 정의 시 다형성(polymorphism)을 위해 포괄 프로시저(generic procedure)를 이용하였기 때문에 동일한 메소드를 통해 헤더의 특정 키, 헤더 전체, 또는 트레이스 자료에 모두 접근할 수 있다. 효율성을 위해 헤더만 읽거나 복사하는 등의 경우가 아니라면 일반적인 작업은 SuTrace 클래스의 객체를 이용한다.

SuTrace 클래스와 SuHeader 클래스는 메모리에 존재하는 SU 파일을 다루기 위한 클래스인 반면에, SuFile 클래스는 디스크에 존재하는 SU 파일을 메모리에 읽어들이거나 메모리에 존재하는 파일을 디스크에 쓰는데 사용한다. SuFile 클래스의 객체에서는 read 메소드를 통해 SuHeader 또는 SuTrace 객체를 생성할 수 있고, write 메소드를 이용하여 SuTrace 객체를 파일로 출력할 수 있다. SuFile 클래스의 객체는 SuFile 생성자(constructor)

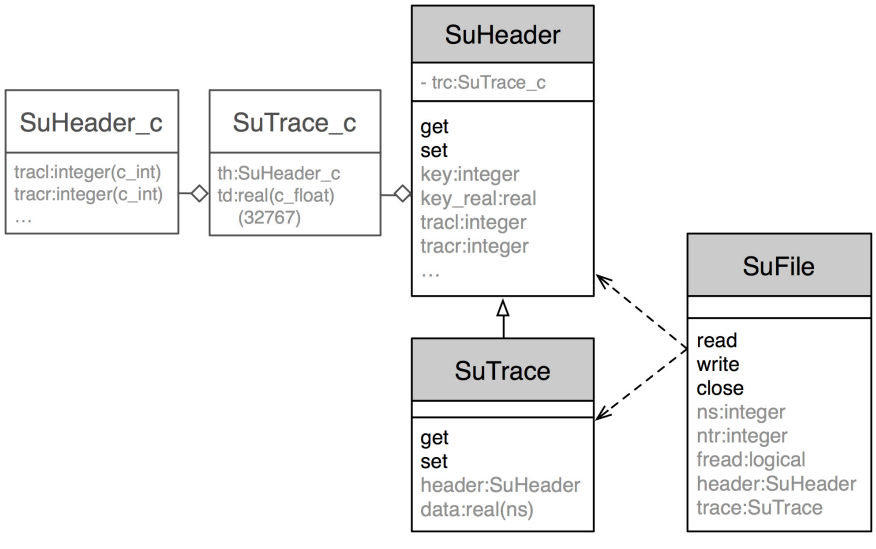

Fig. 4. A class diagram of the object-oriented Fortran library. The top box in each class shows the class name. The second box shows important member variables, and the third box shows method names. Methods in gray are redundant functions for easy use. Type names after colons indicate the type of member variables or the return type of methods. Methods without a return type are (polymorphic) subroutines.

함수나 su_input 및 su_output 함수를 이용하여 생성할 수 있다. 이렇게 생성된 객체는 내부적으로 C언어 입출력을 위해 파일 위치에 대한 참조 정보와 입력을 위한 파일인지 출력을 위한 파일인지에 대한 정보, 시스템 고유 자료형인지 XDR 자료형인지 등의 정보를 포함한다. 그러나, 이러한 정보는 사용자가 직접 접근할 필요가 없으므로, 정보 은닉(encapsulation)을 통해 접근할 수 없도록 제한해두었다. 그 외에 입력의 경우 미리 샘플 개수 정보를 저장해두고, 표준 입력이 아닌 파일 입력을 사용할 경우 파일 내에 존재하는 총 트레이스의 개수에 대한 정보도 저장해두는데, 이러한 정보는 사용자가 함수를 통해 접근할 수 있다.

사용 예

본 라이브러리를 이용하면 사용자가 만든 프로그램에서 SU 파일을 바로 읽어 들여 자료 처리를 수행하거나 자료처리 결과를 SU 파일로 출력할 수 있다. 본 라이브러리를 이용한 일반적인 자료 처리 과정은 다음과 같다. 우선 입력 SU 파일과 출력 SU 파일에 해당하는 SuFile 객체들을 생성하며 파일을 연다. 이후 입력 파일의 첫 번째 트레이스부터 반복적으로 읽어 들이면서 SuTrace 객체를 생성한다. SuTrace 객체를 이용하여 자료 처리를 수행하고 난 후에는 출력 파일에 처리된 SuTrace 객체를 넘겨 파일에 쓴다. 모든 트레이스에서 작업이 끝나면 SuFile 객체들을 통해 파일을 닫는다.

위와 같이 일종의 필터 형식으로 사용하는 프로그램 외에도, 사용자가 만든 파동 전파 모델링 프로그램에서 공통 송신원 모음을 바로 SU 파일로 출력하게 만들 수 있다. 구조보정 프로그램에서는 SU 형식의 공통 송신원 모음을 읽어 들이고 구조보정 영상을 SU 파일로 출력할 수 있다. 또한 파형 역산 프로그램에서는 역산 결과 파일을 SU 형식으로 출력할 수도 있다. SU 파일 형식으로 출력하게 되면 Seismic Un*x 패키지에 있는 다양한 프로그램들을 별도의 수정 없이 바로 사용할 수 있다.

Table 4와 5는 Seismic Un*x에 존재하는 suwind와 sustrip 및 supaste 명령을 포트란으로 구현한 것이다. 이 프로그램들은 이미 명령어가 존재하므로 작성할 필요가 없지만 예시로 구현하였다. Table 4에서는 표준 입력 SU 파일에서 “fldr” 키 값이 10 이상 20 이하인 트레이스만 추출하여 표준 출력으로 내보낸다. 이해를 돕기 위해 코드를 설명하면, 1행에서 프로그램을 시작하고, 2행에서 SU 파일 입출력 모듈을 불러온다. 3행에서 입력 및 출력 SU 파일에 접근하기 위한 SuFile 객체들을 선언하고, 4행에서 SU 트레이스를 저장하기 위한 SuTrace 객체를 선언한다. 5행과 6행에서 사용할 키와 키의 최소 및 최대 값을 선언한다. 8행과 9행에서 표준 입력과 표준 출력을 통해 SU 파일을 참조하는 SuFile 객체들을 생성한다. 10행부터 13행의 루프에서는 입력 파일(sf)의 끝에 도달할 때까지 반복적으로 SU 트레이스를 하나씩 읽어 들여 SuTrace 객체(trc)에 저장한다. 11행에서는 SuTrace 객체에서 원하는 키 값(keyval)을 읽어 들이고, 12행에서는 읽어 들인 키 값이 원하는 범위에 속할 경우 출력 파일(so)에 SU 트레이스를 쓴다. 입력 파일의 끝에 도달하여 위의 루프가 끝나면 14행에서 입출력 SuFile 객체들을 제거하여 SU 파일에 대한 참조를 마치고 15행에서 프로그램을 끝낸다. 예시에서는 가독성을 높이기 위해 클래스 이름에 대소문자를 구분하였으나, 포트란 언어는 문자열이 아닌 코드의 대소문자를 구분하지 않는다. 이

Table 4. A Fortran program which mimics the “suwind” command

```
1   program suwind
2   use suio
3   type(SuFile):: sf, so
4   type(SuTrace):: trc
5   character(len=4):: keyword=“fldr”
6   integer:: keymin=10, keymax=20, keyval
7
8   sf = su_input()
9   so = su_output()
10  do while( sf%fread(trc) )
11      keyval = trc%key(keyword)
12      if(keymin <= keyval .and. keyval <= keymax) call so%write(trc)
13  enddo
14  call su_close(sf, so)
15  end program
```

프로그램은 suwind 명령어를 이용하면

```
suwind key=fldr min=10 max=20 < input.su > output.su
```

와 같이 수행할 수 있다.

Table 5에서는 “input_header.su” 파일에서 트레이스 헤더를 읽고, “input_data.su” 파일에서 트레이스 자료를 읽어 “output.su” 파일에 적는다. 이 과정은 샘플 개수가 1000개일 때 Seismic Un*x 명령어를 이용하면 다음과 같다.

```
sustrip head=header < input_header.su > /dev/null
sustrip < input_data.su > data.bin
supaste ns=1000 head=header < data.bin > output.su
```

본 라이브러리를 사용하면 Seismic Un*x 패키지에서 제공하지 않는 기능도 수행할 수 있다. 예를 들면, Table 6은 XDR 형식의 SU 파일을 표준 입력으로부터 읽어 시스템 고유의 이진 형식으로 출력하는 프로그램으로, suoldtonew의 역변환 프로그램이다. 특수한 경우로, IEEE 754 부동소수점 형식을 사용하는 리틀엔디안 시스템에서는 시스템 고유 이진 형식과 XDR 형식 사이에 엔디안만 다르므로 suswapbytes 명령어로 동일한 결과를 얻을 수 있다. Table 7의 경우 메모리에 존재하는 속도모델을 SU 파일로 출력하는 프로그램으로, 주시 토모그래피나 파형 역산과 같이 속도 모델을 출력하는 경우에 사용할 수 있다. Table 7의 경우 헤더 키 값들과 속도 모델을 별도의 이진 파일로 출력한 후 suaddhead와 sushw 명령어를 통해 동일한 결과를 얻을 수도 있으나, 본 라이브러리를 이용하면 간편하게 바로 SU 형식의 속도 파일을 얻을 수 있다.

Table 5. A Fortran program which mimics the “supaste” command

```
 1  program supaste
 2  use suio
 3  type(SuFile):: sh, sd, so
 4  type(SuTrace):: trc
 5  integer:: itr, ntr, ns
 6
 7  sh = su_input(“input_header.su”, ntr=ntr)
 8  sd = su_input(“input_data.su”, ns=ns)
 9  so = su_output(“output.su”)
10  do itr = 1, ntr
11       trc = sd%trace(itr)
12       call trc%set(header=sh%header(itr), ns=ns)
13       call so%write(trc)
14  enddo
15  call su_close(sh, sd, so)
16  end program
```

Table 6. A Fortran program which convert an XDR-formatted SU file to a native-binary-formatted SU file

```
 1  program suNewToOld
 2  use suio
 3  type(SuFile):: sf, so
 4  type(SuTrace):: trc
 5
 6  sf = su_input(xdr=.true.)
 7  so = su_output(xdr=.false.)
 8  do while( sf%fread(trc) )
 9       call so%write(trc)
10  enddo
11  call su_close(sf,so)
12  end program
```

Table 7. A Fortran subroutine which generates an SU file from a velocity array

```
1   subroutine vel_to_su(fout, vel, n1, n2, d1, d2)
2   use suio
3   character(len=*), intent(in):: fout
4   integer, intent(in):: n1, n2
5   real, intent(in):: d1, d2, vel(n1,n2)
6   integer:: itr, scalco=100
7   type(SuFile):: sf
8   type(SuTrace):: trc
9
10  trc = SuTrace(ns=n1, dt=nint(d1), scalco=-scalco, ntr=n2, d1=d1, d2=d2) ! same for all traces
11  sf = su_output(trim(fout))
12  do itr = 1, n2
13      call trc%set(tracl=itr, gx=nint((itr-1)*d2*scalco), data=vel(:,itr))
14      call sf%write(trc)
15  enddo
16  call sf%close()
17  end subroutine
```

결 론

본 연구에서는 포트란 언어를 사용하여 탄성파 자료처리를 수행하는 연구자들과 학생들을 위해 Seismic Un*x의 SU 파일을 읽고 쓸 수 있는 라이브러리를 개발하였다. 기본 SU 파일 입출력 함수들은 XDR 형식과 표준 입출력 지원을 위해 C 언어로 구현하였고, 이러한 함수들은 포트란 2003의 ISO_C_BINDING 모듈을 이용하여 포트란 함수들과 연결하였다. 사용자들이 사용하게 될 포트란 프로그래밍 인터페이스는 포트란 2003에 도입된 객체 지향 설계를 이용하여 개발하였다. 파일을 다루는 SuFile 클래스, 트레이스를 다루는 SuTrace 클래스, 필요할 경우에는 트레이스 헤더만을 다루는 SuHeader 클래스를 이용해 사용자가 파일 입출력 및 트레이스 접근이 가능하도록 하였다. 본 라이브러리를 사용하면 포트란 언어를 사용하여 쉽게 SU 파일을 읽고, 수정하고, 쓸 수 있을 것이다. 본 연구를 통해 개발한 라이브러리와 설명서는 https://github.com/pkgpl/SUIO에서 찾을 수 있다.

사 사

이 논문은 부경대학교 환경해양대학 간접비 2013년도 신진교수 연구력 강화 지원 사업, 산업통상자원부 자원개발특성화대학 사업 및 한국과학기술정보연구원/슈퍼컴퓨팅센터로부터 지원을 받아 수행된 연구임(KSC-2014-C1-015).

References

Barry, K.M., Cavers, D.A. and Dneale, C.W., 1975, “Report on recommended standards for digital tape formats”, *Geophysics*, Vol. 40, No. 2, pp. 344-352.

Clapp, R.G., Prucha, M.L., Sava, P., Dellinger, J. and Biondi, B., 2004, *SEP Manual*, Stanford Exploration Project, p. 7.

IEEE Computer Society, 2008, *IEEE Standard for Floating-Point Arithmetic*, Institute of Electrical and Electronics Engineers, p. 1.

International Association of Oil & Gas Producers, 2012, *OGP P1/11 Geophysical position data exchange format*, International Association of Oil & Gas Producers, p. 1.

Madagascar, 2015.1.21., http://www.ahay.org.

Metcalf, M., Reid, J. and Cohen, M., 2012, Modern Fortran Explained, Oxford, pp. 243-286.

Metz, S., 2013, Practical Object-Oriented Design in Ruby, Pearson Edication, Inc., pp. 1-14.

SEG Technical Standards Committee, 2002, *SEG Y rev 1 Data Exchange format*, Society of Exploration Geophysicists, p. 1.

SEG Technical Standards Committee, 2006, *Shell Processing Support Format for 3D Surveys, Rev 2.1*, Society of Exploration Geophysicists, p. 1.

SEG Technical Standards Committee, 2012, *SEG-D, Rev 3.0 SEG Field Tape Standards*, Society of Exploration Geophysicists, p. 1.

Sheen, D.-H., Ji, J. and Lee, D.-S., 2003, “Development

of a PC-based 3-D seismic data processing system for the underground investigation", *Journal of the Korean Society for Geosystem Engineering*, Vol. 40, No. 1, pp. 48-57.

Stockwell, J.W. and Cohen, J.K., 2008, *The New SU User's Manual*, Center for Wave Phenomena, p. 21.

Stockwell, J.W., 1999, "The CWP/SU: Seismic Un*x package", *Computers & Geosciences*, Vol. 25, pp. 415-419.

Sun Microsystems, Inc., 1987, XDR: External Data Representation Standard, Network Working Group, p. 1.

UKOOA Exploration Committee, 1990, *UKOOA P1/90 Post Plot Data Exchange Tape 1990 Format*, UKOOA Exploration Committee, p. 1.

Xiang, Y., Cuzzocrea, A., Hobbs, M. and Zhou, W., 2011, *Algorithms and Architectures for Parallel Processing*, Springer, pp. 433-442.

하 완 수

현재 부경대학교 에너지자원공학과 조교수
(本 學會誌 第51券 第5号 參照)